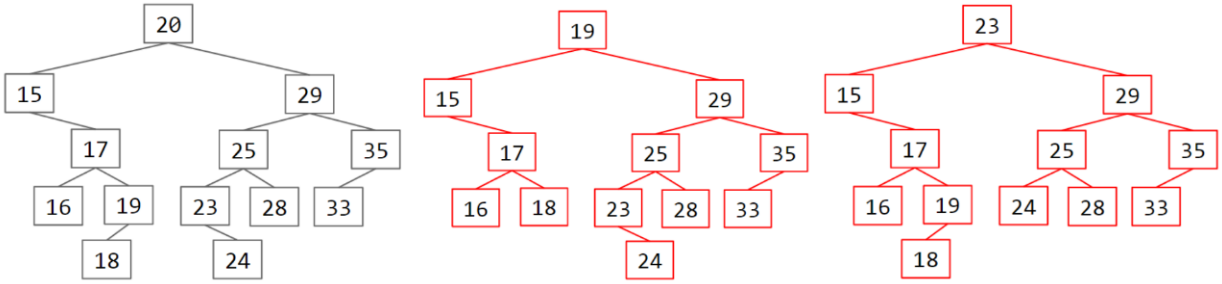
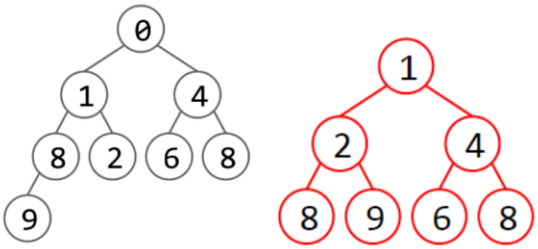
CS 61B Discussion 10 Exam Prep Spring 2017

1 Basic Operations (Spring 2015 MT2 Q1)

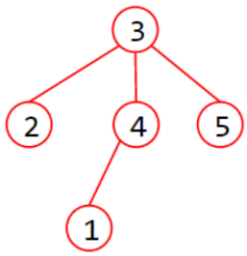
a. **To the right of the BST below**, draw a BST that results if we delete 20 from the BST. You should use the deletion procedure discussed in class (i.e. no more than 4 references should change).
Either of the two trees in red below are correct. In the left case, we've chosen the predecessor of 20, i.e. 19, as the new root. In the right, we've chosen the successor.



b. **To the right of the minHeap below**, draw the minHeap that results if we delete the smallest item from the minHeap.
The only correct answer if we use the procedure from class is the minHeap shown tothe right.



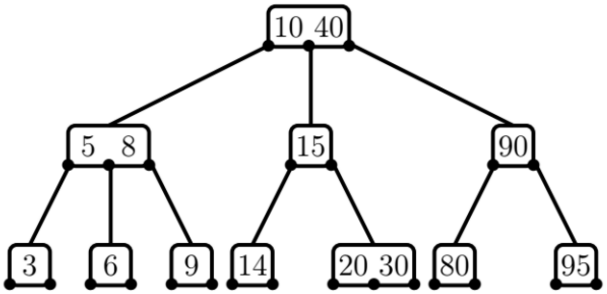
d. Draw a valid Weighted Quick Union object that results after the following calls to connect: `connect(1, 4)`, `connect(2, 3)`, `connect(1, 3)`, `connect(5, 1)`. Don't worry about the order of the arguments to each connect call, we'll accept any reasonable convention.
There are many correct answers depending on your convention for order of arguments. The most important points are that when you connect items, that you're only connecting the roots of each subtree to other roots, and that when you connect 5 to the weight 4 tree, that the 1 points directly at the root of that tree. One example solution is given below. Giving an answer in terms of an id[] array was also acceptable.



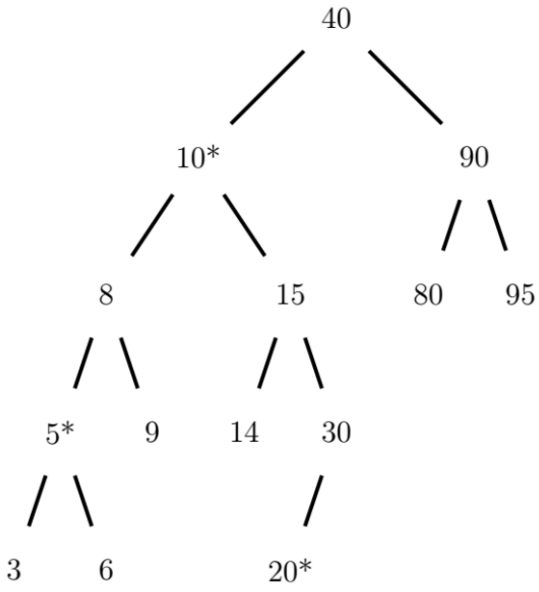
2 (Fall 2016 Final Q6)

a. Show the left-leaning red-black tree that corresponds to the (2,3) tree on the left. Indicate red nodes with an asterisk (as in part (b) below).

(2, 3) tree

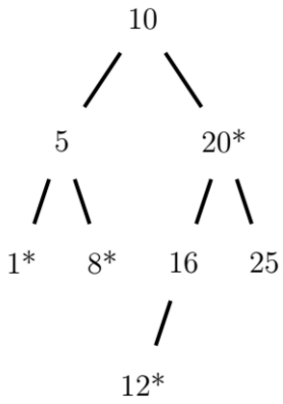


Left-Leaning Red-Black Tree

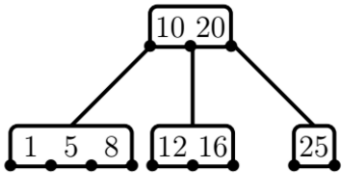


b. Show the (2,4) tree that corresponds to the red-black tree on the left. Red nodes are marked with an asterisk.

Red-Black Tree

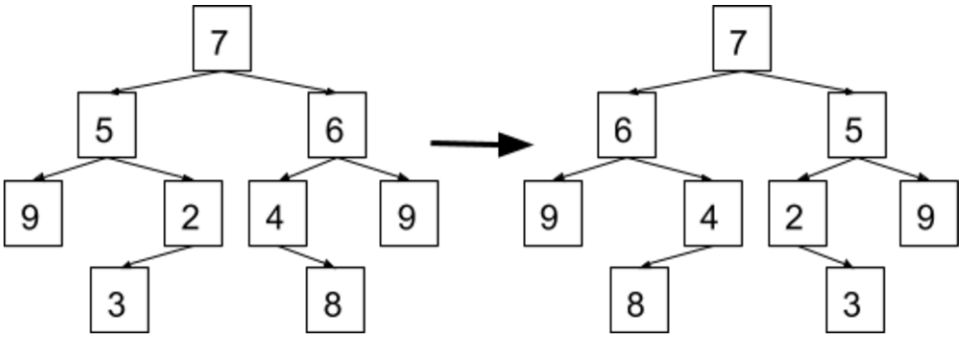


(2, 4) tree



3 Lapras (Summer 2016 MT2 Q7)

Fill in a method, `Tree::flipHorizontally`, which should flip a **symmetric** binary tree's values destructively about the root in linear time. Some helper methods (`swapNumbers` and `safePush`) are given. You may not define your own helper methods. See the example:




```
public class Tree {
    private TreeNode root;

    private static class TreeNode {
        private int num;
        private TreeNode left, right;

        private TreeNode(int num, TreeNode left, TreeNode right) {
            this.num = num;
            this.left = left;
            this.right = right;
        }
    }

    private static void swapNumbers(TreeNode t1, TreeNode t2) {
        int temp = t1.num;
        t1.num = t2.num;
        t2.num = temp;
    }

    private static void safePush(TreeNode t, Stack<TreeNode> s) {
        if (t != null) {
            s.push(t);
        }
    }

    // Continues on next page
```

```
    public void flipHorizontally() {
        // Line breaks included for clarity

        Stack<TreeNode> forwardsTraversal = new Stack<>();
        Stack<TreeNode> backwardsTraversal = new Stack<>();

        safePush(root.left, forwardsTraversal);
        safePush(root.right, backwardsTraversal);

        while (!forwardsTraversal.isEmpty()) {

            TreeNode first = forwardsTraversal.pop();
            TreeNode second = backwardsTraversal.pop();

            swapNumbers(first, second);

            safePush(first.right, forwardsTraversal);
            safePush(first.left, forwardsTraversal);
            safePush(second.left, backwardsTraversal);
            safePush(second.right, backwardsTraversal);
        }
    }
}
```